Cof C

OIPE JC98 FEB 16 2005 PATENT & TRADEMARK OFFICE

Docket: 4208-4003

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s)	:	Olkkonen et al.			
Serial No.	:	09/891,382	Art Unit	:	2665
Filed	:	December 10, 1997	Examiner	:	T. Tran
US Patent No.:		6,842,460	Issue Date	:	Jan 11, 2005
For	:	AD HOC NETWORK DISCOVERY MENU			

REQUEST FOR CERTIFICATE OF CORRECTION OF PATENT

Mail Stop: -
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Certificate
FEB 2 2 2005
of Correction

Sir:

Pursuant to 35 U.S.C. § 254 and 37 CFR § 1.322, Patentees hereby request that the attached Form PTO-1050, in duplicate, be approved and that a Certificate of Correction be issued. At least one of the attached copies of the Form PTO-1050 is suitable for printing. No Certificate of Correction fee is believed due. The Commissioner is, however, authorized to charge any fees that may be required by this paper to **Deposit Account 13-4500, Order No. 4208-4003**.

The above-identified United States Patent (6,842,460 ("the '460 patent")) matured from Non-Provisional application Serial Number 09/891,382, filed June 27, 2001. A review of the '460 patent reveals that the United States Patent and Trademark Office (USPTO) erred in printing the

FEB 2 3 2005

patent as discussed below.

ERRORS IDENTIFIED

On the cover sheet of the patent, the following non-patent literature prior art reference, cited by Patentees, was not identified:

> "Miller et al., "BLUETOOTH REVEALED" 2001 Prentice Hall PTR; pp. 164-176 and 217-222".

DISCUSSION

On June 27, 2001, Patentees filed US application Serial No. 09/891,382, which ultimately matured into the '460 patent on the Issue Date of January 11, 2005. Accompanying the original application papers, was a first Information Disclosure Statement ("the first IDS"), which included a first Form PTO-1449 Information Disclosure Citation ("the first PTO-1449") citing five separate pieces of prior art, i.e., three US patents (5,822,309; 6,134,587; and 6,104,712), one international publication (WO 01/37497), and the non-patent literature publication identified in the previous section. A second Information Disclosure Statement ("the second IDS"), which included a second Form PTO-1449 Information Disclosure Citation ("the second PTO-1449") and cited three additional US Patents, i.e., 6,304,556; 6,411,815; and 6,459,894), was filed by Patentees on March 13, 2001. Each of the first IDS and the second IDS included copies of the cited prior art documents.

On October 23, 2003, the Examiner mailed a first non-final Office Action that included copies of the first PTO-1449 and the second PTO-1449, each initialed, signed, and dated by the Examiner. The copy of the first PTO-1449 for reference citation "AR" (the above publication to

FEB 2 3 2005

Miller et al.) under the "OTHER DOCUMENTS" section, however, was neither initialed or crossed out by the Examiner. A copy of the first PTO-1449 that accompanied the non-final Office Action is attached hereto. Also attached, for the convenience of the USPTO, is an additional copy of the subject Miller et al. publication for which Patentees are again seeking citition via a Certificate of Correction

On January 15, 2004, Patentees filed a timely Response and Request for Reconsideration that, in response to the non-final Office Action that included, in the "REMARKS" section on Page 10, a request that the Examiner "consider Reference AR listed on the Form PTO-1449 filed on June 27, 2001", i.e., the first PTO-1449. A second request for consideration of the "AR" reference citation on the first PTO-1449 was also included in Patentees Amendment and Request for Reconsideration filed June 22, 2004 in response to the final Office Action of April 7, 2004.

Notwithstanding Patentees' two separate requests for consideration of reference "AR" cited on the first PTO-1449, the '460 patent issued without the identification of the Miller et al. publication (Reference "AR") under INID Code (56), in a subsection titled "OTHER PUBLICATIONS", on the cover page of the '460 patent. As such, it would appear that the Examiner erred by (1) sending out an incomplete copy of the Examiner initialed, dated, and signed first Form PTO-1449 with the non-final Office Action, and (2) not subsequently correcting such error in response to Patentees' repeated requests for correction thereof.

FEB 23 2005

CONCLUSION

Accordingly, Patentees respectfully submit that the above-identified error was not caused by Patentees, and hereby request that a Certificate of Correction be issued adding the above-identified publication to Miller et al., i.e., "Reference AR" identified on the first FORM PTO-1449 filed with the first IDS of June 27, 2001.

AUTHORIZATION

The Commissioner is hereby authorized to charge any fee deficiencies under 37 CFR § 1.20, or to credit any overpayment, to Deposit Account No. 13-4500, Order No. 4208-4003. A duplicate copy of this Request is enclosed for this purpose.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: February 16, 2005

By: [illegible]
Brian W. Brown
Registration No. 47,265
(202) 857-7887 Telephone
(202) 857-7929 Facsimile

CORRESPONDENCE ADDRESS:
MORGAN & FINNEGAN, L.L.P.
Three World Financial Center
New York, New York 10281-2101
(212) 758-4800
(212) 751-6849 Facsimile

FEB 23 2005

Serial No.: 09/891,382
Request for Certificate of Correction

O I P E
FEB 16 2005
JC98
PATENT & TRADEMARK OFFICE

Docket: 4217-4003

CONCLUSION

Accordingly, Patentees respectfully submit that the above-identified error was not caused by Patentees, and hereby request that a Certificate of Correction be issued adding the above-identified publication to Miller et al., i.e., "Reference AR" identified on the first FORM PTO-1449 filed with the first IDS of June 27, 2001.

AUTHORIZATION

The Commissioner is hereby authorized to charge any fee deficiencies under 37 CFR § 1.20, or to credit any overpayment, to Deposit Account No. 13-4500, Order No. 4208-4003. A duplicate copy of this Request is enclosed for this purpose.

Respectfully submitted,
MORGAN & FINNEGAN, L.L.P.

Dated: February 16, 2005

By: [illegible]
Brian W. Brown
Registration No. 47,265
(202) 857-7887 Telephone
(202) 857-7929 Facsimile

CORRESPONDENCE ADDRESS:
MORGAN & FINNEGAN, L.L.P.
Three World Financial Center
New York, New York 10281-2101
(212) 758-4800
(212) 751-6849 Facsimile

57702 v1

FEB 23 2005

OIPE JC98 FEB 16 2005 PATENT & TRADEMARK OFFICE

J1036 U.S. PTO 09/891382 06/27/01

↑AFFIX CUSTOMER NO. LABEL ABOVE ↑

Page 1 of 1

FORM PTO-1449 **INFORMATION DISCLOSURE CITATION**	Attorney Docket: 4208-4003	Serial No.: Unassigned
	Applicant: Olkkonen et al.	
	Filing Date: Herewith	Group Art Unit: Unassigned

U.S. PATENT DOCUMENTS

Examiner Initial		Document Number	Date	Name	Class	Sub-Class	Filing Date
TT	AA	5,822,309	10/13/98	Ayanoglu et al.			
TT	AB	6,134,587	10/17/00	Okanoue			
TT	AC	6,104,712	08/15/00	Robert et al.			
	AD						
	AE						
	AF						
	AG						
	AH						
	AI						
	AJ						
	AK						

FOREIGN PATENT DOCUMENTS

		Document Number	Date	Country	Class	Sub-Class	Translation
TT	AL	WO 01/37497	05/25/01	WIPO			☐ Yes ☐ No
	AM						☐ Yes ☐ No
	AN						☐ Yes ☐ No
	AO						☐ Yes ☐ No
	AP						☐ Yes ☐ No

OTHER DOCUMENTS (Including Author, Title, Date, etc.)

	AR	Miller et al, "BLUETOOTH REVEALED", 2001 Prentice Hall PTR; pp. 164-176 and 217-222.
	AS	
	AT	

Examiner [illegible]	Date Considered 10/16/03

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP §609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to Applicant.

25617 v1

FEB 23 2005

BLUETOOTH REVEALED

THE INSIDER'S GUIDE TO AN OPEN SPECIFICATION FOR GLOBAL WIRELESS COMMUNICATIONS

BRENT A. MILLER • CHATSCHIK BISDIKIAN

BLUETOOTH REVEALED

BRENT A. MILLER • CHATSCHIK BISDIKIAN

PH PTR

PRENTICE HALL PTR
UPPER SADDLE RIVER, NJ 07458
WWW.PHPTR.COM

Library of Congress Cataloging-Publication Data
Miller, Brent A.
Bluetooth revealed / Brent A. Miller, Chatschik Bisdikian.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-090294-2
1. Bluetooth technology. 2. Telecommunication–Equipment and supplies.
3. Computer Network protocols. I. Bisdikian, Chatschik. II. Title.
TK5103.3 .M55 2000
004.6'20--dc21

Editorial/production supervision: *Kathleen M. Caren*
Acqusition Editor: *Mary Franz*
Editorial Assistant: *Noreen Regina*
Marketing Manager: *Bryan Gambrel*
Manufacturing Manager: *Maura Zaldivar*
Cover Design: *Nina Scuderi*
Cover Illustration: *Tom Post*
Cover Design Director: *Jerry Votta*
Series Design: *Gail Cocker-Bogusz*

The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact
Corporate Sales Department, Prentice Hall PTR
One Lake Street
Upper Saddle River, NJ 07458
Phone: 800-382-3419; FAX: 201-236-7141
Email (Internet): corpsales@prenhall.com

Printed in the United States of America

10 9 8 7 6 5 4 3

ISBN 0-13-090294-2

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Pearson Education Asia Pte. Ltd.
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

In fact the programming model suggested in RFCOMM is a specific instance of the generalized model suggested in the section entitled "The Application Group" in Chapter 5 (refer back to Figure 5.4). In this case we suggest a thin layer of Bluetooth adaptation software for legacy applications that maps platform APIs to specific functions of the Bluetooth protocol stack. In the case of RFCOMM, which provides an emulated serial port, this adaptation software (which the specification calls a port emulation entity) needs to map the application's interactions with a "real" RS-232 serial port to the equivalent operations for the RFCOMM emulated serial port. For the most part these are expected to be initialization operations such as activating and configuring the serial port and establishing a serial connection; and finalization operations such as terminating the serial connection. Once a general serial port adaptation layer is in place in a system, all those legacy applications that use serial communication ought to be enabled to use Bluetooth transports via the RFCOMM emulated serial port.

As pointed out earlier, the use of serial ports is prevalent in devices and environments where Bluetooth wireless communication is likely to be used, and the majority of the version 1.0 profiles depend upon serial port communications. In the absence of a version 1.0 specification for general networking, the RFCOMM protocol provides an important utility for legacy applications. The implementation of this protocol, along with adaptation software for legacy applications that use serial communications, permits many simple cable replacement applications of Bluetooth wireless communication.

The Service Discovery Protocol (SDP)

Service discovery is a process by which devices and services in networks can locate, gather information about and ultimately make use of other services in the network. In traditional networks such as LANs, these services might be statically configured and managed by a network administrator. In these environments, the administrator or end user performs the configuration that is necessary for one participant in the network to use the services of some other network member. For example, a PC user might specify all of the information associated with a network e-mail service (including the mail server name, user and account names, e-mail type, capabilities and options, and so on) to the PC's operating system and applications; once all this information is entered into the PC

MM is a spe-
ction entitled
e 5.4). In this
ıre for legacy
: of the Blue-
ides an emu-
cation calls a
ıctions with a
ɔns for the
are expected
ıfiguring the
:ation opera-
;eneral serial
;acy applica-
to use Blue-

nt in devices
n is likely to
l upon serial
:ification for
n important
ıis protocol,
ıt use serial
applications

ices in net-
nake use of
h as LANs,
y a network
nd user per-
: in the net-
example, a
ı a network
ount names,
s operating
into the PC

and associated with that e-mail service, then the e-mail service becomes available to the PC user.

Administered network services of this sort may be fine for many fixed networks but are really not suitable for temporary mobile networks (ad hoc networks) such as those that might be formed using Bluetooth wireless communication. In these environments a more dynamic, flexible and adaptive solution is needed. Graham, Miller and Rusnak [Graham99] observe the growing incidence of these ad hoc networks and the resulting demand for self-configuring systems:

> The emergence of information appliances and new types of connectivity is spurring a new form of networking: unmanaged, dynamic networks of consumer devices that spontaneously and unpredictably join and leave the network. Consumers will expect these ad hoc, peer-to-peer networks to automatically form within the home, in very small businesses and in networked vehicles. ...
>
> To achieve the goals of simplicity, versatility and pleasurability, the appliances and the network(s) they join must *just work* right *out of the box*. By *just work* we mean that the participants on the network must simply *self configure*. By *self configure* we mean that these network devices and services simply discover each other, negotiate what they need to do and which devices need to collaborate **without any manual intervention**.[5]

Protocols for service discovery can help to enable this self-configuration. Since much of the interdevice communication in Bluetooth usage scenarios is of a peer-to-peer, ad hoc nature, the SIG determined that a service discovery protocol in the stack could provide significant value. The resulting protocol, known as SDP, is a central component of nearly all of the profiles and usage cases, both existing and foreseen.

The service discovery concept is not new or unique to Bluetooth wireless technology. Numerous service discovery technologies are available in the industry, some of them well known. As is evident in other layers of the protocol stack, the SIG is content to adopt existing protocols where it makes sense to do so. In the case of service discovery, though, the SIG developed its own protocol unique to and optimized for Bluetooth wireless communication rather than adopting some other service discovery protocol in the industry. The reasons should become evident as we review SDP's development in the next section.

5. Reprinted by permission from *Discovering Devices and Services in Home Networking*, copyright (1999) by International Business Machines Corporation.

SDP Development

The need for a service discovery component in the protocol stack was recognized early in the process of developing the specification, although direct work on SDP did not commence until later. In early and mid 1998, many of the initial participants in the SIG were focusing on the transport protocols and key middleware protocols like RFCOMM. While the need for other protocols had been identified, task forces of experts to develop these protocols had not been assembled in all cases. In the case of SDP, some preliminary work had been started at Intel and Ericsson in the summer of 1998.

In early internal versions of the specification, service discovery was a section within the L2CAP part of the specification. Initially, L2CAP channels were modeled after a computer bus, and service discovery was concerned exclusively with the transport of Plug and Play parameters over this virtual bus. In September 1998, at a SIG meeting in London,[6] author Bisdikian suggested that the addition of a transport protocol for Plug and Play parameters unnecessarily complicated the L2CAP specification, and that such a protocol merited its own service discovery portion of the Bluetooth specification.

In October 1998 the SIG held a developers conference in Atlanta which author Miller attended. Based upon conversations during that conference, Miller was asked to chair the service discovery task force of the SIG's software working group shortly thereafter. The following month the newly constituted group met for the first time as a formal SDP task force.

While at this time (late 1998) many of the protocol layers had been under development for several months, with some of them approaching levels of stability that would soon near final stages, SDP was still really in a nascent state of forming the requirements and the beginning of a proposal to address those requirements.

Among the identified objectives for Bluetooth SDP were:

Simplicity: Because service discovery is a part of nearly every Bluetooth usage case, it is desirable that the service discovery process be as simple as possible to execute. For the SDP task force this also implied the reuse of other Bluetooth protocols to the extent possible.

6. To advance the development of the specification, face-to-face meetings among SIG members have taken place in many different countries reflective of the multinational constituency of the SIG membership.

FEB 2 3 2005

col stack was
specification,
ater. In early
vere focusing
rotocols like
en identified,
been assem-
ork had been

ice discovery
ion. Initially,
d service dis-
'lug and Play
SIG meeting
of a transport
nplicated the
own service

ice in Atlanta
s during that
task force of
he following
e as a formal

ers had been
approaching
as still really
eginning of a

ere:

arly every
ry process
ce this also
it possible.

g SIG members
nstituency of the

Compactness: As described in previous chapters, the formation of Bluetooth communication links between two devices can in some cases be time consuming. Since service discovery is a typical operation to perform soon after links are established, the SDP air-interface traffic should be as minimal as feasible so that service discovery does not unnecessarily prolong the communication initialization process.

Versatility: The version 1.0 specification includes a number of profiles, and future revisions will undoubtedly add to the list, which is expected to continue to grow. Since an exhaustive set of profiles, usage cases and associated services cannot be foreseen or accurately predicted, it is important for SDP to be easily extensible and versatile enough to accommodate the many new services that will be deployed in Bluetooth environments over time. To support this objective the SDP task force chose a very broad definition of "service," so that the widest possible spectrum of features (services) could be supported in the future.

Service location by class and by attribute: In the dynamic ad hoc networking environment it is important to enable client devices and users to quickly locate a specific service when they already know exactly (or at least largely) what they are looking for. It should be straightforward to search for a general class of service (say, "printer"), for specific attributes associated with that service (for example, "color duplex IBM printer") and even for a specific instance of a service (such as a specific physical printer).

Service browsing: In addition to searching for services by class or attribute, it is often useful simply to browse the available services to determine if there are any of interest. This is a different usage scenario than is searching for specific services, and in some respects it is almost a contrary objective, but the SIG agreed that both usage models have merit, and they developed a solution that uses a consistent method to support both specific service searching and general service browsing.

These objectives led to the development of requirements for a simple, flexible protocol and data representation for service discovery in the protocol stack. Popular industry discovery protocols were reviewed, but none seemed to provide a good match for the SDP objectives—many of these technologies provide robust and comprehensive service discovery and access methodologies, but the SIG was really looking for a fairly low-level, simple, narrow-in-scope solution that met the rather modest objectives noted above in a straightforward manner.[7] At this point Motorola® approached the SIG with a proposal to contribute some technology, suitable for use in Bluetooth service discovery, that Motorola had had under development for several years. Through a contributing adopter agreement Motorola was then able to participate in the SDP task force of the SIG (and in fact the Motorola representative served as editor of the SDP specification), with their contributed technology forming the basis for SDP.

So with the SDP effort underway in November 1998, the SDP requirements and scope were agreed upon and the specification development ensued, incorporating the ideas contributed by Motorola along with the many contributions by the other SIG member companies. Even though the real SDP work started later than for many other protocols, through hard work the SDP specification was completed, ratified and published along with the bulk of the other protocols in the stack in July 1999 in the version 1.0 specification.

The following sections describe some of the key facets of the SDP specification, including why these elements are significant and the rationale for including them in the specification.

SDP Examined

Key to understanding the development of SDP is to understand its motivation and requirements. In fact, this information is included at the beginning of the SDP portion of the specification. As noted above, SDP is intended to allow devices in Bluetooth environments to locate available services. As the specification states, these environments are qualitatively different from traditional networks such as LANs or WANs. Devices and services are likely to come and go frequently in Bluetooth piconets. Thus SDP was developed to satisfy the requirements of such environments.

Some of the notable requirements for SDP are listed in the preceding section. These are also mentioned and expanded upon in the specification. Also of interest are those items that SDP does not attempt to address, at least in version 1.0 of the specification.[8] The "Non-Requirements and Deferred Requirements" portion of the SDP specification

7. Subsequent to publication of the version 1.0 specification, efforts were begun to map some of the leading industry service discovery technologies to the Bluetooth stack. Chapter 16 gives details of this work.
8. Of these, the Bluetooth SIG might choose to enhance SDP in the future to address some of the issues. Many, however, are likely to remain outside the scope of Bluetooth SDP, since some of the issues can be and are addressed by industry discovery protocols, which Bluetooth SDP can accommodate, as explained in the main body text.

o contribute
covery, that
rough a con-
articipate in
presentative
ributed tech-

98, the SDP
cation devel-
torola along
companies.
other proto-
eted, ratified
the stack in

s of the SDP
nd the ratio-

derstand its
cluded at the
above, SDP
locate avail-
s are qualita-
s or WANs.
in Bluetooth
ents of such

the preced-
in the speci-
ot attempt to
Non-Require-
specification

to map some of
apter 16 gives

dress some of the
P, since some of
uetooth SDP can

can be summarized largely with the statement that SDP is narrow in scope, focusing primarily on discovery in Bluetooth environments and leaving more sophisticated service functions and operations to other protocols which might be used in conjunction with SDP.

SDP includes the notion of a client (the entity looking for services) and a server (the entity providing services). Any device might assume either role at a given time, acting sometimes as a service client and sometimes as a service provider (server).

The service provider needs to maintain a list of *service records* that describe the service(s) it provides; this list is called the *service registry*. A service record is simply a description of a given service in a standard fashion as prescribed by the specification. A service record consists of a collection of service attributes containing information about the class of the service (which might be printing, faxing, audio services, information services, and so on), information about the protocol stack layers that are needed to interact with the service, and other associated information such as human-readable descriptive information about the service. Figure 8.3 illustrates the general structure of a service registry with its constituent service records. Shown is a set of services, each with a service record handle (depicted by srvRecHnd[0] through srvRecHnd[j]) and a set of attributes per service (shown as srvAttribute[0:a] through srvAttribute[j:c]). Further explanation of the content of these service records follows.

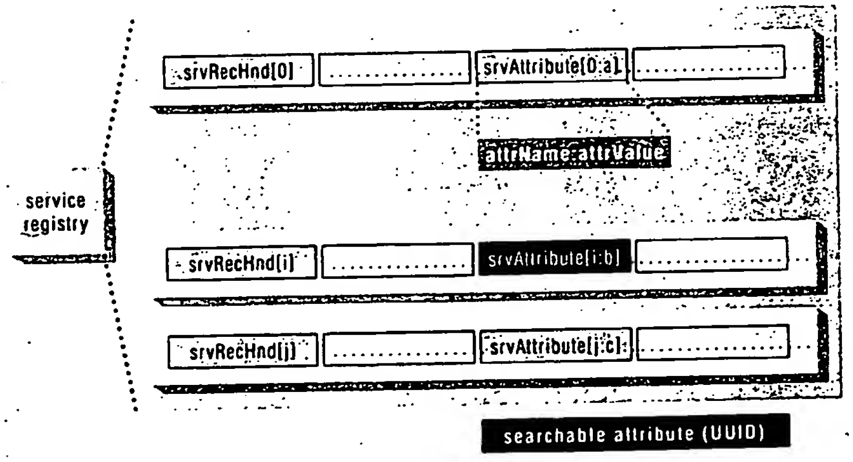


Figure 8.3
General SDP service registry structure.

Service records consist of both ***universal service attributes*** and ***service-specific attributes***. The universal service attributes are simply those parts of the service record that apply to all types of services, such as the service class and protocol stack information noted above. Service-specific attributes are those parts of the service record that are relevant only for a specific class or instance of a service. Examples of service-specific attributes could include attributes specific to a printing service (such as color, duplex and finishing capabilities), attributes specific to an audio service (such as data rate or encoding scheme) or attributes specific to a dial-up networking service (such as serial port configuration or modem setup information). Volume 1 of the specification includes definitions for a set of universal service attributes (those which could apply to all types of services), but it does not include service-specific attributes, since it would be impossible to specify and predict all of the attributes for every imaginable type of service. Service-specific attributes are defined in profiles (volume 2 of the specification). Since profiles describe a usage scenario and how the protocol stack is used, they effectively define a service. So, for example, the headset profile defines the service specific attribute "remote audio volume control" that applies to the headset service. While the universal service attributes can apply to all types of services, this does not mean that they are mandatory–it is not required that every service include every universal service attribute in its service record. In fact, only two of the universal service attributes are mandatory: the service class attribute, which defines the class, or type of the service, and the service record handle, which serves as a pointer, or reference to the service record and is used by the client to access the server's service record.

Each service attribute in a service record consists of an attribute identifier (attribute ID, a 16-bit unsigned integer) and an attribute value associated with that attribute ID. Each entry of the service record is one of these (***attribute, value***) pairs. Because these attributes describe all sorts of information, SDP uses the concept of a ***data element*** for the attribute value. A data element is simply a self-describing piece of data. The first part of a data element consists of a one-byte header that tells the actual type and size of the data. The remainder of the data element consists of the data values for the attribute, of the format and size specified by the data element header. Through the use of data elements, SDP allows attribute values to be of several types, including strings, Booleans, signed and unsigned integers of various sizes, and universally unique identifiers (UUIDs, discussed further below). Moreover, these data types can be lists of the scalar elements noted above, thus providing a

s and *service-*
those parts
h as the ser-
vice-specific
ant only for
vice-specific
ice (such as
to an audio
specific to a
n or modem
s definitions
apply to all
c attributes;
ne attributes
tributes are
nce profiles
l, they effec-
defines the
at applies to
can apply to
ndatory–it is
ice attribute
ce attributes
the class, or
serves as a
the client to

an attribute
ribute value
ecord is one
ribe all sorts
the attribute
ata. The first
lls the actual
nt consists of
cified by the
SDP allows
s, Booleans,
sally unique
, these data
providing a

flexible representation for the many data types of which attribute values might be composed.

Discovering a service in Bluetooth wireless communication reduces to a simple operation: the client specifies the service(s) of interest and the server responds, indicating any available services that match what the client specified. In practice for SDP this consists of the client's sending a request in the form of an SDP protocol data unit (SDP_PDU) that indicates what service(s) it is searching for and the server's sending back a response, also in the form of an SDP_PDU, that indicates what services match the request that the client has made. To accomplish this, the client needs a standard way to represent the service(s) of interest and the server needs a standard method to match its available services against the client's specification. For this purpose SDP introduces *universally unique identifiers* (UUIDs).

A UUID is a concept adopted from the International Organization for Standardization (ISO). UUIDs are 128-bit values that can be created algorithmically and, generally speaking, can be virtually guaranteed to be entirely unique–no other UUID ever created anywhere will have the same value.[9] One advantage of using UUIDs is that new identifiers can be created for new services without requiring a central registry of identifiers maintained by the SIG, although the SIG does include a list of "well-known" UUIDs in the specification for those services related to the published profiles. So a client looking for a service just specifies the UUID associated with that class of service (or with the specific service) in its service search request, and the service provider matches that UUID against those of the services it has available to generate its response.

The SDP_PDUs exchanged between the client and server are simple transactions. The general SDP protocol flow requires only two transactions; the specification defines three different SDP transactions, but the third is really just a composite of the first two. A typical SDP transaction consists of:

1. Client sends a request to search for service(s) of interest; server responds with handles to services that match the request.
2. Client uses the handle(s) obtained in step 1 to form a request to retrieve additional service attributes for the service(s) of interest.

9. This concept is sometimes hard to grasp, but universally unique identifiers can in fact be created. While there is an extremely small chance of duplication, UUIDs as defined by ISO (see [ISO96]) are quite sufficient for the purposes of Bluetooth SDP and turn out to be quite valuable in this context.

Following the above transaction, the client will presumably use the information obtained in step 2 to open a connection to the service using some protocol other than SDP to access and utilize the service. Step 1 is called the *ServiceSearch* transaction and consists of the *ServiceSearchRequest* SDP_PDU from the client to the server and the *ServiceSearchResponse* SDP_PDU in return (from server to client). As noted above, the ServiceSearchResponse SDP_PDU contains handles to one or more services that match the request. In step 2, the client presents one or more of those handles in a *ServiceAttributeRequest* SDP_PDU which causes the server to generate a *ServiceAttributeResponse* SDP_PDU; this exchange is the *ServiceAttribute* transaction. In the ServiceAttributeResponse SDP_PDU will be the attribute values associated with the service that correspond to the attribute IDs that the client specified in the ServiceAttributeRequest SDP_PDU. These attributes may be a combination of universal service attributes and service-specific attributes, and in most cases should provide the client with enough information to subsequently connect to the service.

The specification defines a third SDP transaction, called the *ServiceSearchAttribute* transaction. This transaction consists of a *ServiceSearchAttributeRequest* SDP_PDU from the client to the server followed by a *ServiceSearchAttributeResponse* SDP_PDU from the server to the client. It is actually redundant to the first two transactions described above and is included for efficiency. What the ServiceSearchAttribute transaction allows is the combination of steps 1 and 2. That is, the client can form a single request that specifies not only services to search for but also the attributes to return for matching services in the server's response. The server then responds with handles to matching services as well as the requested attribute values for those matching services. An implementer thus has a choice between the two alternatives for SDP transactions.[10] More importantly, though, the ServiceSearchAttribute transaction may in some cases be more efficient in terms of the number of bytes transmitted over the air-interface. The consolidated transaction itself requires more bytes than the individual transactions but could result in fewer total transactions. Especially in cases where many service records are being accessed, such as in a service browsing application, the ServiceSearchAttribute transaction might be more efficient.

10. It should be noted, however, that different profiles mandate the use of different SDP transactions, so if a profile is being implemented, the profile will determine which SDP transaction(s) need to be used, and the programming effort to support all three transactions should not be great.

FEB 23 2005

ably use the
ervice using
ce. Step 1 is
viceSearchRe-
erviceSearch-
oted above,
ne or more
ents one or
PDU which
_PDU; this
AttributeRe-
vith the ser-
cified in the
be a combi-
ributes, and
ation to sub-

led the *Ser-*
a *ServiceS-*
followed by
the client.
l above and
transaction
can form a
out also the
ponse. The
well as the
nplementer
nsactions.[10]
action may
bytes trans-
ction itself
ld result in
ice records
on, the Ser-

SDP transac-
' transaction(s)
ould not be

In a nutshell, this is most of what is needed for SDP transactions. The specification also includes protocol definitions for special cases, including an error response SDP_PDU and a mechanism, called the *continuation state*, for dealing with server responses that cannot fit into a single SDP_PDU.[11] The syntax of these protocol transactions and the data elements that they carry is detailed in the specification and is not reproduced here. A unique feature of the SDP chapter of the specification is the inclusion of several detailed protocol examples as an appendix to the SDP specification. The members of the service discovery task force of the SIG who developed the specification felt that because the actual byte streams generated for SDP transactions can be complex (even though the transactions themselves are conceptually simple), it would be useful to include the examples as a guide for implementers. The complexity is introduced mostly when complex data elements (such as *DataElementSequences*, which are lists of data elements and which can be nested) are carried in the SDP_PDUs. When these complex data types are included in SDP_PDUs, or when SDP_PDUs need to be split using the continuation state information, the various "count" fields that introduce segments of the SDP_PDUs need to accurately reflect the number of bytes that follow in that segment. The examples in the specification serve to clarify the correct construction of SDP_PDUs.[12]

Figure 8.4 summarizes the SDP transactions. Shown in the figure are representations of the relevant arguments and parameters passed in the SDP_PDUs, although these are not complete lists of all arguments and parameters; the complete syntax is in the specification. As Figure 8.3 shows, only services and service attributes that are described by UUIDs are searchable. Attributes of a service which are not described by UUIDs are not searchable and can be retrieved only after a service has been located using a UUID attribute.

11. The client can specify the maximum size for the response to its request SDP_PDU. It is possible for the response that is generated by the server to be larger than this maximum size. In this case, the server includes some continuation state information at the end of its response, which allows the client to initiate another request to obtain the next portion of the response, if desired.
12. In fact the developers of the specification learned first hand of the need for these examples when they constructed them, since there were some errors in the first internal versions of the examples. There were even some errors in the examples published in the original version 1.0A SDP specification, which were subsequently corrected in version 1.0B.

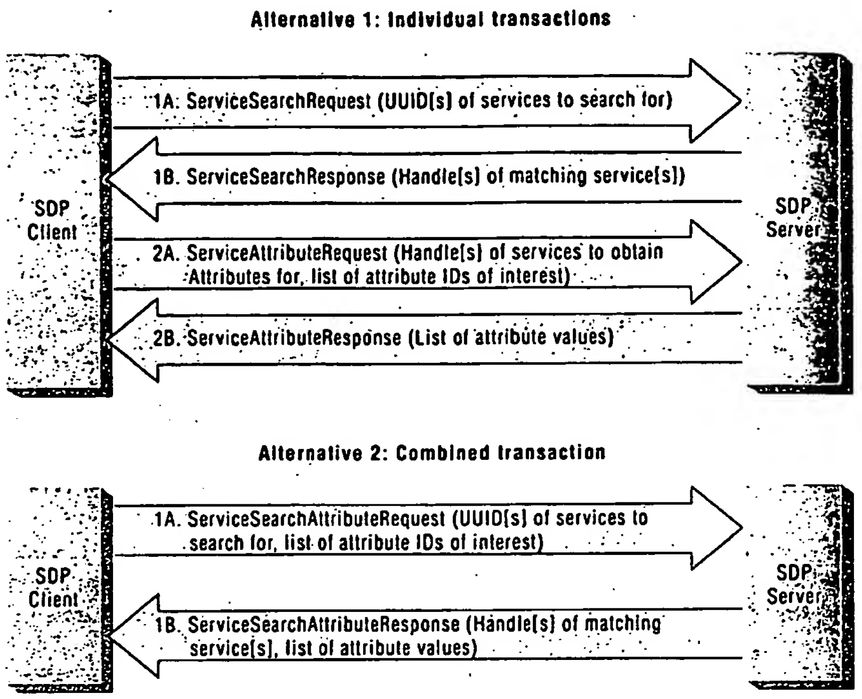
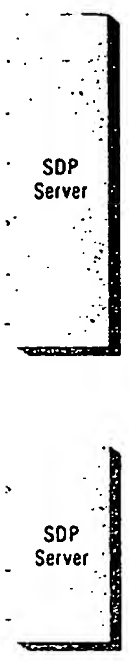

Alternative 1: Individual transactions
SDP Client
SDP Server
1A. ServiceSearchRequest (UUID[s] of services to search for)
1B. ServiceSearchResponse (Handle[s] of matching service[s])
2A. ServiceAttributeRequest (Handle[s] of services to obtain Attributes for, list of attribute IDs of interest)
2B. ServiceAttributeResponse (List of attribute values)
Alternative 2: Combined transaction
SDP Client
SDP Server
1A. ServiceSearchAttributeRequest (UUID[s] of services to search for, list of attribute IDs of interest)
1B. ServiceSearchAttributeResponse (Handle[s] of matching service[s], list of attribute values)

Figure 8.4
SDP transaction summary.

SDP Usage

Since SDP was developed primarily for discovering services in Bluetooth environments, the applications most likely to make use of SDP will be those developed specifically to be aware of Bluetooth wireless communication (as opposed to legacy applications). One exemplary application for SDP usage is what we will call the *Bluetooth Piconet Minder*[13] (or BPM application). Such an application is likely to be included, in one form or another, in many Bluetooth devices. A BPM application as we envision it would present a view of available devices and services in proximity (in a piconet) to the user and to other applications. This could include a user interface; one might imagine icons or other representations of devices and services. Such an application could

13. This term is used generically here and is not known to, or intended to, conflict with any actual product names.



:s in Blue-
se of SDP
:h wireless
exemplary
ıoth Piconet
:ely to be
·s. A BPM
ıle devices
er applica-
ıe icons or
ıtion could

ith any actual

give a user a central point to manage the Bluetooth connections to other devices and to select and make use of the services offered by those other devices. To support such functions, a BPM application might make use of service searching and service browsing and thus initiate SDP transactions to populate the service information that is exposed to the user and to other applications.

Certainly other applications designed for use in Bluetooth environments might use SDP. Every profile (or at least every "non-generic" profile that involves concrete usage scenarios) includes an SDP service record to be used when implementing that profile. Applications written specifically to exercise the protocol stack will probably need to execute SDP transactions to successfully instantiate the profiles. First, such applications will need to execute SDP transactions with another device to determine if that device offers the desired service, and if so, those applications will need to execute additional SDP transactions to obtain the information from the service record about how to access that service (this can include such information as the required protocol stack and associated parameters that the service uses).

In the case where multiple applications use SDP (perhaps one or more profile applications and a BPM application), it may be advantageous to implement a central SDP client and SDP server that are available to all the applications that need them. These SDP "helper applications" could be implemented as part of the common services layer that was described in Chapter 5. The applications could use the platform APIs to access the common SDP services which would generate the SDP transactions and pass the retrieved information back to the applications.

The Service Discovery Application Profile (SDAP) detailed in Chapter 12 offers guidance for application interactions with SDP. While, as previously noted, the specification does not define APIs, the SDAP does define primitive operations that could be mapped to APIs and events on many platforms, thus providing a basis for SDP common services.

There may be legacy applications that make use of service discovery, but such applications probably use some other industry discovery protocol (perhaps Jini™, Universal Plug and Play™, Salutation™, Service Location Protocol, the IrDA service discovery protocol, or some other protocol). Since SDP was developed for Bluetooth applications, legacy applications would not be expected to include this protocol without modification to the application. Even for these applications, though, SDP does offer some accommodation. One of the design points for SDP was to ensure that other popular industry discovery protocols could be used in conjunction with it. One of the things that can be discovered using SDP is that the service supports one or more other discovery protocols. Thus SDP might be used in the initial service discovery phase to locate the service; further SDP transactions might be used to discover that the service supports, say, Salutation; once this has been determined, the newly discovered protocol (Salutation in the example) can be used for further interaction with the service. SDP specifically supports this sort of operation. A SIG white paper [Miller99] describes how Salutation can be mapped to SDP. Similar mappings to other technologies should be possible, and the SIG is working toward formalizing some of these mappings as profiles, as discussed in Chapter 16.

equest signaling
Chapter 7. This
for Bluetooth
ce can be raised
e requests from

e places a secu-
it does not ini-
transport and
th which it is to
curs before the
te PDU, as dis-

ecurity mode at
mode 3 cannot
uthenticates all
le security archi-
9]. It forms the

ed with activities
stimuli (such as
), the idle mode
hese procedures
e discovery and

scover devices in
e device discov-
dly name of dis-
g the name could
lving the hosts.
purpose of creat-
for future use. In
communications
bonding, a device
f creating a bond
nsactions.
service discovery
ote devices. This

procedure follows the guidelines for service discovery found in the SDAP, which is presented next.

The Service Discovery Application Profile

As noted in Chapter 8, service discovery is expected to be a key component of most Bluetooth applications. Nearly all of the profiles include a service discovery element. Like the GAP, the Service Discovery Application Profile, or SDAP, provides a common and standard method for performing service discovery using the Bluetooth protocol stack.

Unlike most other profiles, the SDAP describes a standard service discovery application model and also defines abstractions of service primitives that in some respects resemble application programming interfaces (APIs). Even though the SDAP deals with the SDP middleware layer protocol and thus addresses some of the "invisible" operations described earlier, it is aimed primarily at application writers. It is the only profile with "application" in its title and the only profile to suggest API-like primitives. As explained in Chapter 8, these primitives could be mapped to platform APIs in a straightforward manner.

SDAP Development

Both authors have a special interest in the SDAP. Author Bisdikian served as editor of the SDAP portion of the specification, conceived the original idea for the SDAP and contributed most of its content, and author Miller chaired the service discovery task force responsible for delivering the SDAP.

As with the GAP, the need for an SDAP was not originally evident, and thus the SDAP was also developed late in the specification cycle. Not until January 1999, when most profiles were already underway, was the question raised regarding whether or not a service discovery profile was needed. By March of that year the idea of an application profile for service discovery was accepted and the SDAP development proceeded.

The development of the SDAP is rooted in the fundamental assumptions that led to the formation of the SIG itself: the diversity and number of devices that would be capable of Bluetooth wireless communication and the diversity and number of services available through these devices would steadily increase. To keep a semblance of order in the expected sea of devices and services available to a user, it was recognized that a standardized procedure should be created that would allow the user of a device to locate and identify services.

The SDAP does not describe how the service discovery itself is performed; it relies on SDP for this task. Rather, SDAP describes how an application that uses SDP should be created and should behave. In particular, it defines the functional characteristics of such an application through a set of service primitive abstractions, detailed below. Furthermore, the SDAP defines how other profiles and applications in general must use the group of Bluetooth transport protocols to carry SDP_PDUs when they need to execute SDP transactions. This latter item was an expansion of the SDAP's original scope. All of the "nongeneric" profiles contain an SDP section that provides a list of parameters for the protocol stack that leads to the particular application covered by the profile. These protocol stack parameters include ones like the RFCOMM data link connection identifier (DLCI) needed to reach, say, the PPP layer in the LAN access profile. These parameters are carried as service attributes within SDP_PDUs. However, these other profiles do not specify how the SDP layer could use the group of Bluetooth transport protocols to carry these SDP_PDUs. Since the latter process should be identical for all profiles, one idea was to include it in a generic profile like the GAP. However, the GAP does not focus on the transport of data with a source or sink above the L2CAP layer. Moreover, the SDP specification itself does not contain the dependencies of SDP on the group of Bluetooth transport protocols. Even though this may seem like an oversight, it was a deliberate choice. The Bluetooth service discovery protocol, although tied to systems that utilize Bluetooth wireless communication, is in principle a transport-independent protocol. Hence, the SDP specification focuses exclusively on the SDP transactions themselves and the various SDP_PDUs that are used, as well as the type and form of information that is carried in them. The SDP specification does not particularly focus on how these transactions are carried over the Bluetooth air-interface. Ultimately, SDAP became the only (and natural) point of reference for describing how the SDP layers use the group of Bluetooth transport protocols to carry the SDP_PDUs to each other.

SDAP Examined

The SDAP is unique among the application-oriented profiles, like the file transfer profile, the LAN access profile, and so on. These other profiles describe how the complementary parts of a user-level application

:ted that would

:covery itself is
describes how
uld behave. In
: an application
below. Further-
:ions in general
cols to carry
ns. This latter
of the "nonge-
t of parameters
ion covered by
ones like the
d to reach, say,
:ers are carried
: other profiles
p of Bluetooth
: latter process
nclude it in a
ot focus on the
P layer. More-
ependencies of
en though this
The Bluetooth
at utilize Blue-
rt-independent
ly on the SDP
at are used, as
in them. The
se transactions
SDAP became
how the SDP
: to carry the

ofiles, like the
iese other pro-
/el application

running on two (or more) devices work together to support a particular usage scenario. The application in SDAP needs to be present in only one device. This application interacts with the SDP layer of the stack in the device where it resides to initiate SDP interactions with one or more SDP layers in other devices, so as to learn about services in those other devices. Upon the arrival of responses from the other devices, the service discovery application can make those results available to the user of the device that initiated the transaction(s).

Often, service discovery in non-Bluetooth environments is performed by broadcasting inquiries for services or inquiries for locating directories of services. In the latter case, when a service directory is found, it is then contacted to find out about services that are registered there. In a Bluetooth piconet, broadcasts are entirely unidirectional in that they are exclusively directed from the master to the slaves of the piconet. Furthermore, broadcast transmissions are not recoverable in that they cannot be retransmitted following an error in their transmission. Thus, service discovery in a Bluetooth piconet does not use a broadcast model. Service discovery in Bluetooth piconets is closely associated with device discovery. Service discovery is executed only between fully identified pairs of devices and only after they have discovered each other and have created a Bluetooth link (up to and including an L2CAP connection) between them.

According to the SDAP, devices participating in service discovery may have either of the following roles:

- *Local device:* This device implements the service discovery application, like the service browsing application referred to in Chapter 8. It also implements the client portion of the SDP layer. A local device initiates SDP transactions as shown in Figure 8.3.
- *Remote device:* This device is contacted by a local device to inquire about services. A remote device implements the server portion of the SDP layer. It responds to SDP transaction requests from a local device. To produce its responses, the remote device maintains, explicitly or implicitly, a database[4] that contains service records for the services available via the remote device.

4. The specification does not define the format of this database, leaving that choice to implementers. Moreover, this need not be a "database" in the classic sense; it is simply a collection of information maintained by the SDP server in a format suitable for the device.

Even though some devices may act only as local or as remote devices, these device roles are, in general, temporary and meaningful only when an SDP transaction between two devices is under way. A device can be a local or remote device at different times or even at the same time, depending upon when it creates service inquiries or responds to them. The SDAP device roles bear no relation to the baseband roles of master and slave, as the latter roles are meaningless above the link manager layer. A local device could be either a master or a slave in its piconet, as could a remote device.[5]

The SDAP is the only profile that uses the term application in its title. However, the SDAP does not define any particular application. Such a definition would be very much platform dependent and possibly too restrictive for application developers, neither of which is a desirable objective for a specification. However, the SDAP specifies the services that a service discovery application should provide to its users to be useful. These services are summarized in four service primitive abstractions. These primitives could be mapped to an appropriate set of APIs based upon the underlying software and/or hardware platform in which an SDAP application is instantiated. An example mapping of these primitives to the Salutation APIs is given in [Miller99]. These primitives are:

- *serviceBrowse:* This service primitive is utilized when a local device wants to perform general service searches, referred to in Chapter 8 as service browsing. These searches might take the form of queries about what services in general or what services of type S are available, if any, via a selected set of remote devices. This application-level service primitive results in SDP_PDU transactions initiated by any one of the three basic request SDP_PDUs presented in Chapter 8: *SDP_ServiceSearchRequest*, *SDP_ServiceAttributeRequest*, or *SDP_ServiceSearchAttributeRequest*. Optionally, the *LMP_name_request* PDU could also be sent to learn the user-friendly name of the remote device.
- *serviceSearch:* This service primitive is utilized when a local device wants to perform searches for a specific type of service. The search could take the form of queries about what services of type *S* with attributes A1 and A2 are available, if any, via a selected set of remote devices. Similar to the previous primitive,

5. Often a local device acts as master of a piconet, since typically it would be the device that desires to create connections with other devices and search for and use services on them. However, this does not mean that a local device must be a master to perform service inquiries. A slave device could equally well initiate such inquiries.

d or as remote
and meaningful
s under way. A
s or even at the
ce inquiries or
ion to the base-
aningless above
r a master or a

pplication in its
ılar application.
ent and possibly
ch is a desirable
fies the services
users to be use-
ive abstractions.
t of APIs based
m in which an
; of these primi-
primitives are:

l when a local
s, referred to in
might take the
what services of
remote devices.
in SDP_PDU
e basic request
viceSearchRequest,
hAttributeRequest.
also be sent to
e.

l when a local
type of service.
what services of
le, if any, via a
vious primitive,

e the device that
rvices on them. How-
service inquiries. A

this application-level service primitive results in SDP_PDU transactions initiated by any one of the three basic request SDP_PDUs previously mentioned.

- *enumerateRemDev:* This service primitive is used when a local device wants to search for remote devices in its vicinity. The search can be restricted with a class of device (CoD) qualifier to search only for devices belonging to the specified class. This application-level service primitive results in the local device executing inquiries to learn about devices, primarily any new devices, in its vicinity. If, in the future, CoD-specific *dedicated inquiry access codes* (DIACs) are standardized, then the inquiries for this primitive could be generated according to these DIACs. Otherwise, the *generic inquiry access code* (GIAC) is used and the local device needs to filter the received responses according to the information included in the CoD field in the FHS BB_PDUs received from the responding devices.
- *terminatePrimitive:* This service primitive results in the termination of the operations invoked by the previous primitives.

The first and second primitives above relate directly to transactions involving SDP_PDUs. The third primitive could be satisfied merely by requesting that the device enter the inquiry mode with the sole purpose of searching for any other devices in the inquiry scan state (as described in the baseband section of Chapter 6). The last primitive simply terminates any ongoing actions resulting from the use of any of the other primitives.

The SDAP requirements on the Bluetooth stack are straightforward. To implement this profile one needs to use nothing beyond the default settings for all the protocol layers below the SDP layer. In particular, devices that connect for the sole purpose of performing service discovery do not need to authenticate each other or encrypt their Bluetooth link.[6] The SDP transactions are carried over the ACL baseband link between the devices. At the L2CAP layer SDP transactions are carried over connection-oriented channels configured to carry "best effort" traffic.

An additional distinction of this profile is that it identifies the conditions under which L2CAP channels carrying SDP traffic are torn down. This is so because SDP does not define a session or a transport

6. This does not imply that device authentication and/or link encryption should not be performed. This statement simply implies that the SDAP imposes no security precautions for its execution. Security is as much a usage scenario requirement as a user-level settable policy, as discussed in the GAP. Security precautions may still be taken for reasons outside the scope of SDAP.

protocol for carrying SDP_PDUs. SDP itself is in essence a connectionless protocol. To run the connectionless SDP request/response transactions, the L2CAP layer needs to maintain the L2CAP channel that carries the transactions at least for the duration of the transaction. For efficient use of the transmission resources, the L2CAP channel between an SDP client and an SDP server should be maintained even longer. Ultimately, it is the application on behalf of which SDP transactions are executed that should open and maintain, for as long as necessary, the L2CAP channel for SDP transactions between two devices.

Summary

In this chapter we have highlighted the two generic Bluetooth profiles: the GAP and the SDAP. The GAP describes the connectivity and security modes of operation for a device that permits it to discover, be discovered by, and create trust bonds and Bluetooth links with other devices. These modes of operation are user- (or application-) settable device policies that specify how the device should behave relative to other devices with which Bluetooth communication might ensue.

The SDAP describes the functional characteristics of a service discovery application. Furthermore, and equally importantly, the SDAP describes the way that the SDP layer must use the group of Bluetooth transport protocols to carry SDP transactions. This aspect of the SDAP also forms the basis for executing service discovery within the other profiles.

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NUMBER	:	6,842,460 B2
DATED	:	January 11, 2005
INVENTOR(S)	:	Olkkonen et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the cover page

Under INID Code (56), References Cited, please insert:

"OTHER PUBLICATIONS

Miller et al., "BLUETOOTH REVEALED" 2001 Prentice Hall PTR; pp. 164-176 and 217-222."

MAILING ADDRESS OF SENDER: Morgan & Finnegan, LLP
3 World Financial Center
New York, NY 10281-2101

PATENT NO. 6,765,353 B2
No. of additional copies
⇒

57716 v1

FEB 23 2005

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NUMBER	:	6,842,460 B2
DATED	:	January 11, 2005
INVENTOR(S)	:	Olkkonen et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the cover page

Under INID Code (56), References Cited, please insert:

"OTHER PUBLICATIONS

Miller et al., "BLUETOOTH REVEALED" 2001 Prentice Hall PTR; pp. 164-176 and 217-222."

MAILING ADDRESS OF SENDER: Morgan & Finnegan, LLP
3 World Financial Center
New York, NY 10281-2101

PATENT NO. 6,765,353 B2

No. of additional copies

⇒

57716 v1

FEB 23 2005